## Problema 5

## Implementação

## espectro.m

```
% Entradas:
% A - matriz (n x n)
% Saídas:
% S = vetor (n \times 1) ortogonal
function [S] = espectro(A, tol)
  % Definição de variáveis
  erro = tol + 1;
  S = diag(A);
  while tol <= erro
    % Processo iterativo
    [Q, R] = qr_GSM(A);
    A = R * Q;
    % Verificação de convergência
    novo S = diag(A);
    erro = norm(S - novo_S,'inf');
    S = novo_S; % Atualiza o resultado
  end
end
```

## **Testes**

Para os testes, gero matrizes com números inteiros uniformemente distribuídos entre 1 e 9. A matriz é então multiplicada por sua transposta para que seja simétrica e, portanto, tenha autovalres reais. Então comparamos os autovalores obtidos pela função criada com os autovalores obtidos pela função eig do MATLAB.

```
>> M = randi(9,5,5);
>> M = M' * M;
>> flip(eig(M))
    606.58   61.75   29.118   4.9587   0.58848
>> S = espectro(M, 1e-12)
    606.58   61.75   29.118   4.9587   0.58848
```

Os testes acima (assim como todos os posteriores) foram feitos com uma tolerância de  $10^{\{-12\}}$ , e mesmo assim foram obtidos muito rapidamente.

Uma ideia interessante para escalar os testes para matrizes maiores, é verificar a norma entre a diferença do resultado das funções, ao invés de comparar os vetores diretamente.

```
>> N = randi(9,10,10);
>> N = N' * N;
>> S = espectro(N, 1e-12);
>> S - flip(eig(N))
   9.0949e-13
   -8.5265e-14
   1.4211e-14
   -1.4211e-14
   5.6843e-14
   3.5527e-14
   -1.3603e-11
   1.3443e-11
   -3.1974e-14
   1.9054e-13
>> norm(espectro(N,1e-12) - flip(eig(N)))
   1.9148e-11
```

Verificando para uma matriz de ordem 100, temos:

```
>> 0 = randi(9,200,200)

>> 0 = 0' * 0;

>> norm(espectro(0,1e-12) - flip(eig(0)))

7.5866e-10
```

Esse resultado foi bem precismo, mas a função já demorou bem mais para convergir (aproximadamente 1 minuto ).

Para matrizes maiores (de ordem 300 por exemplo), a nossa função passa a ser muito lenta (demorando quase 6 minutos para convergir) o que torna o seu uso inviável.